

DATA PROCESSOR, PROGRAM UPDATING METHOD AND
STORAGE MEDIUM

BACKGROUND OF THE INVENTION

5 Field of the Invention

The present invention relates to a data processor,
a program updating method and a storage medium, and
more specifically a data processor, a program updating
method and a storage medium which update a program with
10 data sent from outside.

Related Background Art

In a data processor which executes various kinds
of application programs such as those for personal data
management functions, it is general to write a program
15 such as firmware for obtaining executing environments
for the application programs into a nonvolatility
memory.

The program written in the nonvolatility memory is
rewritten when a bug exists in the program or when a
20 version of the program is to be upgraded.

Japanese Patent Application Laid-Open No. 6-44064
discloses a method to rewrite the program stored in the
nonvolatility memory. This method is configured by
receiving and temporarily holding an updating program
25 sent from outside by way of a broadcast wave or a
telephone network, and automatically rewriting the
program stored in the nonvolatility memory into the

received program.

Furthermore, digital broadcast has a high possibility to provide new kinds of services one after another as compared with conventional analog broadcast.

5 Accordingly, a receiver for the digital broadcast which is newly purchased may soon be out-of-date when its functions remain unchanged from those at a time of purchase and cannot cope with the new kinds of services.

10 In order to solve such a problem, there is a concept to modify a software program to cope with the new kinds of services without changing any hardware of the receiver. That is, there is a method which sends a program in a condition overlapped with a broadcast wave
15 for downloading, thereby updating a program in an instrument which is capable of receiving the digital broadcast.

However, the method disclosed by Japanese Patent Application Laid-Open No. 6-44064 is configured by
20 rewriting the program stored in the nonvolatility memory into a new program and may update the program incompletely when a power supply is intercepted due to power failure during the updating of the program, thereby resulting in a situation where a system cannot
25 start up due to the incomplete updating of the program.

Furthermore, the method does not take an operating condition of the instrument into consideration for

downloading and may update the program even while the receiving instrument is operating, thereby resulting in an erroneous operation of the instrument.

5 SUMMARY OF THE INVENTION

A primary object of the present invention is to provide a data processor, a program updating method and a storage medium which are capable of preventing a system from being started up improperly due to
10 incomplete program updating.

Another object of the present invention is to provide a data processor, a program updating method and a storage medium which update a program on the basis of an operating condition on a receiving side.

15 In order to attain these objects, the data processor according to the present invention is a data processor operating on the basis of a program stored in first memory means, and comprising receiving means which receives an updating program sent from outside,
20 comparing means which compares a version of the program stored in the first memory means with a version of the updating program, and control means which stores the updating program into second memory means different from the first memory means when the comparing means
25 judges that the version of the updating program is newer than the version of the program stored in the first memory means.

Furthermore, the data processor according to the present invention is a data processor having processing means which processes data transmitted from outside on the basis of a program stored in the first memory means and outputs the data to an output device, and comprising receiving means which receives an updating program sent from outside, control means which stores the received updating program into second memory means different from the first memory means, and changing means which changes processing by the processing means on the basis of the program stored in the first memory means to processing by the processing means on the basis of the updating program stored in the second memory means.

Furthermore, the data processor according to the present invention is a data processor having processing means which processes data sent from outside on the basis of a program stored in memory means and outputs the data to an output device, and comprising judging means which judges whether or not processing is executed by the processing means, receiving means which receives an updating program sent from outside, memory control means which stores the received updating program into the memory means when the processing is not executed by the processing means and a processing control means which controls the processing means on the basis of the updating program stored in the memory

means.

Moreover, the program updating method according to the present invention is a program updating method which updates a program in a data processor operating on the basis of a program stored in first memory means, and is configured to receive an updating program sent from outside using receiving means, compare a version of the program stored in the first memory means with a version of the updating program and store the updating program into second memory means different from the first memory means when the comparison indicates that the version of the updating program is newer than the version of the program stored in the first memory means.

Moreover, the program updating method according to the present invention is a program updating method which updates a program in a data processor having processing means for processing data sent from outside on the basis of a program stored in first memory means and outputting the data to an output device, and is configured to receive an updating program sent from outside using receiving means, store the received updating program into the second memory means different from the first memory means and change processing by the processing means on the basis of the program stored in the first memory means to processing by the processing means on the basis of the updating program

stored in the second memory means.

Moreover, the program updating method according to the present invention is a program updating method which updates a program in a data processor having
5 processing means for processing data sent from outside on the basis of a program stored in memory means and outputting the data to an output device, and is configured to judge whether or not processing is executed by the processing means, store an updating
10 program sent from outside into the memory means when the processing is not executed by the processing means and control the processing means on the basis of the updating program stored in the memory means.

In addition, the storage medium according to the
15 present invention is a memory means which is to be used in a data processor operating on the basis of a program stored in first memory means, and comprises a step to receive an updating program sent from outside using receiving means, a step to compare a version of the
20 program stored in the first memory means with a version of the updating program and a step to store the updating program into second memory means different from the first memory means when the comparison indicates that the version of the updating program is
25 newer than the version of the program stored in the first memory means.

In addition, the storage medium according to the

present invention is a storage medium which is to be used in a data processor having processing means for processing data sent from outside on the basis of a program stored in first memory means and outputting the data to an output device, and comprises a step to receive an updating program sent from outside using receiving means, a step to store the received updating program into second memory means different from the first memory means and a step to change processing by the processing means on the basis of the program stored in the first memory means to processing by the processing means on the basis of the updating program stored in the second memory means.

In addition, the storage medium according to the present invention is a storage medium which is to be used in a data processor having processing means for processing data sent from outside on the basis of a program stored in memory means and outputting the data to an output device, and comprises a step to judge whether or not processing is executed by the processing means, a step to store an updating program sent from outside into the memory means when the processing is not executed by the processing means and a step to control the processing means on the basis of the updating program stored in the memory means.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram showing a configuration of a first embodiment of the data processor according to the present invention;

5 FIG. 2 is a diagram illustrating status transition of processes to update a program in the data processor shown in FIG. 1;

FIG. 3 is a diagram illustrating status transition of processes to update a program in a second embodiment of the data processor shown in FIG. 1;

10

FIG. 4 is a block diagram showing a configuration of a third embodiment of the data processor according to the present invention;

FIG. 5 is a diagram illustrating status transition of processes to update a program in the data processor shown in FIG. 4;

15

FIG. 6 is a block diagram showing a configuration of an IRD (integrated receiver decoder) for digital broadcasting preferred as a fourth embodiment of the present invention;

20

FIG. 7 is a flowchart illustrating data processings by the IRD preferred as the fourth embodiment;

25

FIG. 8 is a flowchart illustrating data processings by the IRD preferred as the fourth embodiment;

FIG. 9 is a flowchart illustrating data

processings by the IRD preferred as the fourth embodiment;

FIG. 10 is a flowchart illustrating data processings by an IRD preferred as a fifth embodiment
5 of the present invention;

FIG. 11 is a flowchart illustrating data processings by the IRD preferred as the fifth embodiment of the present invention; and

FIG. 12 is a diagram showing a screen for program
10 updating.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Now, the preferred embodiments of the present invention will be described with reference to the
15 accompanying drawings.

(First embodiment)

FIG. 1 is a block diagram showing a configuration of a data processor preferred as the first embodiment of the present invention.

20 The data processor comprises, as shown in FIG. 1, an MPU 10 which controls a system according to a program stored in a flash memory A 12 or a flash memory B 13, an EEPROM 11 which stores a jump instruction which indicates whether MPU 10, after a reset process
25 is released, selects the program stored in the flash memory A 12 or the flash memory B 13, a RAM 14 which provides a work area for the MPU 10 and an external

interface 17 serving as an interface with external devices (not shown): the MPU 10 being connected to each of the blocks by way of a common bus 18. Needless to say, other members are connected as occasion demands.

A common bus 18 has addresses of 20 bits: addresses FFFFOh through FFFFFh assigned to the EEPROM 11, addresses 00000h through 3FFFFh assigned to the flash memory A 12, addresses 40000h through 7FFFFh assigned to the flash memory B 13 and addresses 80000h through BFFFFh assigned to the RAM 14.

Then, description will be made of processings performed by the MPU 10 in the data processor with reference to FIG. 2.

FIG. 2 is a diagram showing status transition of processes to update a program in the data processor shown in FIG. 1.

The processes are carried out on the basis of a program stored in the flash memory A 12 or B 13 and includes a processing to update the program.

When a reset process is released by turning on a power source with a power switch (not shown)(step S210), the MPU 10 designates the addresses FFFFOh and reads out contents of EEPROM 11 assigned to this area. Since a jump instruction written in the contents, the MPU 10 selects the program stored in the flash memory A 12 or B 13 on the basis of the jump instruction and

starts up the system according to the selected program
(step S211). Let us assume here that the flash memory
A 12 is selected and the MPU 10 executes the processing
according to the program stored in the flash memory A
5 12.

In a condition where the system is started up as
described above (step S212), the MPU 10 receives a
program transmitted by a communicating medium such as a
broadcast wave or a telephone network from an external
10 device by way of an external interface 17 (step S213),
checks whether or not the program is completely
received on the basis of a check sum or the like (step
S214) and erase the received program when the reception
of the program is not normally completed (step S221).
15 The MPU 10 terminates the processing in this way (step
S222).

When the reception of the program is normally
completed, the received program is temporally stored in
a RAM 14 (step S215).

20 Then, the MPU 10 judges, on the basis of data such
as ID contained in the received program, whether or not
the received program applies to the system and whether
or not a version of the received program is newer than
that of a currently used program (step S216). When the
25 received program does not apply to the system or when
the version of the received program is not newer than
that of the currently used program, the MPU 10 judges

the received program as an unwanted program and erases it (step S221).

When the received program applies to the system and its version is newer than that of the currently
5 used program, the MPU 10 judges that the received program is a program to be updated and checks the program for its operation (step S217).

To check the received program for its operation, the MPU 10 starts it up and checks its operation with
10 an automatic operation check program. When the MPU 10 recognizes that the received program does not operate normally as a result of the operation check, it erases the received program (step S221).

When the MPU 10 confirms that the received program
15 operates normally as a result of the operation check, it writes the received program i.e. the updating program into the flash memory B 13 which is not selected (step S218) and operates the system in accordance with contents of the program stored in the
20 flash memory B 13 (step S219). When the system does not operate normally, the MPU 10 terminates the processing in a condition where the program is written in the flash memory B 13 (step S222). When the system operates normally, the MPU 10 rewrites the contents of
25 the EEPROM 11 so that the flash memory B 13 is selected when the power source is turned on the next time (step S220) and terminates the processing (step S222).

When the received program is a program to be updated and can normally operates as described above, the received program is written into the flash memory B 13 and the flash memory B 13 is selected when the power source is turned on the next time, whereby a program having a new version is started up when the power source is turned on the next time.

When the system does not operate normally in accordance with the contents of the flash memory B 13, the flash memory A 12 is selected when the power source is turned on the next time, whereby the system operates in accordance with the contents of the flash memory A 12. Accordingly, the data processor is capable of preventing the system from being misoperated or not started due to incomplete program updating into the flash memory B 13.

When the flash memory B 13 is selected, the updating program is written into the flash memory A 12, whereby a program having a newest version is stored in the flash memory A 12 and a program having an older version is stored in the flash memory B 13.

(Second embodiment)

Now, the second embodiment of the present invention will be described with reference to FIG. 3.

FIG. 3 shows status transition of processes to update a program in the second embodiment of the data processor shown in FIG. 1. The second embodiment uses

members which are similar to those of the first embodiment and not described in particular.

The second embodiment is configured to store an automatic operation check program and a jump instruction in the EEPROM 11, store programs having different versions in the flash memory A 12 and the flash memory B 13, check the programs for their operations to judge whether or not the programs operate abnormally before executing the program in a flash memory selected in accordance with the jump instruction, and select the other flash memory when the program in the selected flash memory operates abnormally to execute the program stored in the other flash memory.

When the reset process is released by turning on the power source (step S310), the MPU 10 starts up the automatic operation check program of the EEPROM 11 and checks contents of flash memory designated by the jump instruction stored in the EEPROM 11 (step S311) as shown in FIG. 3. Assuming that the flash memory A 12 is selected, the MPU 10 checks contents of program stored in the flash memory A 12 with the automatic operation check program. When the MPU 10 confirms that the program operates normally with the automatic operation check program, the MPU 10 operates the system in accordance with the contents of the program stored in the flash memory A 12 (step S313) and terminates the

processing (step S316).

When the MPU 10 confirms that the selected program does not operate normally with the automatic operation check program, the MPU 10 checks contents of the flash memory B 13 with the automatic operation check program (step S312) and, when it confirms that the program does not operate normally, it terminates the processing (step S316).

After confirming that the program stored in the flash memory B 13 operates normally, the MPU 10 rewrites contents of the EEPROM 11 so that the flash memory B 13 is selected when the power source is turned on the next time (step S314), operates the system in accordance with the contents of the program stored in the flash memory B 13 (step S315) and terminates the processing (step S316).

(Third embodiment)

Now, the third embodiment of the present invention will be described with reference to FIGS. 4 and 5.

FIG. 4 is a block diagram showing a configuration of a data processor preferred as the third embodiment and FIG. 5 shows status transition of processes to update a program in the data processor shown in FIG. 4.

The data processor preferred as the third embodiment comprises, as shown in FIG. 4, an MPU 40 which executes process control including system control in accordance with a program stored in a ROM 41, a

flash memory 42 which stores a version upgrading
program for upgrading a version of the program stored
in the ROM 41, a RAM 44 which provides a work area for
the MPU 40, an auxiliary memory device 46, a memory
5 interface 45 serving as an interface for the auxiliary
memory device 46 and an external interface 47 serving
as an interface for external devices (not shown): the
MPU 40 being connected to each of the blocks by way of
a common bus 48. Needless to say, other members are
10 connected as occasion demands.

The program stored in the ROM 41 (system program)
is a factory-shipped program. Together with the system
program, an automatic operation check program is stored
in the ROM 41.

15 Processing executed by the MPU 40 in the data
processor will be described with reference to FIG. 5.

When a reset process is released by turning on a
power supply with a power switch (not shown) (step
S507), the MPU 40 reads out the automatic operation
20 check program stored in the ROM 41 and starts up this
program to check contents of the flash memory 42 with
the automatic operation check program (step S508).
Since the flash memory 42 does not store a version
upgrading program until it is taken from outside, the
25 MPU 40 starts up the system with the program stored in
the ROM 41 (step S511).

When the system is started up and the MPU 40

receives a program transmitted by a communicating medium such as a broadcast wave or a telephone network from an external device by way of the external interface 47 during operation of the system (step S513), the MPU 40 checks whether or not the reception of the program is completed on the basis of a check sum (step S514) and when the reception of the program is not completed normally, it erases the received program (step S521) and terminates the processing (step S522).

10 When the reception of the program is completed normally, in contrast, the MPU 40 temporarily stores this program into the RAM 44 (step S515). Then, the MPU 40 judges whether or not the received program is applicable to the system of the data processor and has a version which is newer than that of a currently operating program on the basis of data such as ID contained in the received program (step S516).

15 When the received program is not applicable to the system of the data processor or when the version of the received program is not newer than the version of the currently operating program, the MPU 40 judges this received program as an unwanted program and erases it (step S521).

25 When the received program is applicable to the system of the data processor and has a version newer than that of the currently operating program, the MPU 40 judges that the received program is a version

upgrading program and checks the program for its operation (step S517). For this operation check, the MPU 40 starts up the received program and checks its operation with the automatic operation check program.

5 When the MPU 40 confirms that received program does not operate normally as a result of the operation check, it erases the received program (step S521).

When the MPU 40 confirms that the received program operates normally as a result of the operation check,
10 the MPU 40 writes the received program as a version upgrading program into the flash memory 42 (step S518) and operates the system in accordance with the program stored in the flash memory 42 (step S519). When the system does not operate normally, the MPU 40 terminates
15 the processing in a condition where the program is written in the flash memory 42 (step S522). When the system operates normally, the MPU 40 writes the received program into the auxiliary memory device 46 by way of the memory device interface 45 (step S520) and
20 terminates the processing (step S522).

When the received program is a version upgrading program which operates normally, the received program is written into the flash memory 42 and the auxiliary memory device 46 respectively. Since the version
25 upgrading program stored in the flash memory is selected when the power source is turned on the next time, the data processor starts up the new version

program.

When the power source is turned on the next time (step S507), the MPU 40 reads out the automatic operation check program from the ROM 41, starts up this
5 program and checks operation of the version upgrading program stored in the flash memory 42 with the automatic operation check program (step S508). When the MPU 40 confirms that the version upgrading program operates normally with the automatic operation check
10 program, it operates the system program stored in the ROM 41, thereby starting up a system having an upgraded version (step S511).

When the MPU 40 confirms that the version upgrading program does not operate normally with the
15 automatic operation check program, in contrast, the MPU 40 reads out the version upgrading program from the auxiliary memory device 46 by way of the memory device interface 45 and writes it into the flash memory 42 (step S509). The MPU 40 operates the system program
20 stored in the ROM 41 and the version upgrading program stored in the flash memory 42, thereby starting up the system having the upgraded version (step S511). The MPU 40 subsequently repeatedly executes operations similar to those described above.

25 Since the version upgrading program is stored into both the flash memory 42 and the auxiliary memory device 46 as described above, the MPU 40 can read out

the version upgrading program stored in the auxiliary memory device 46 and start up the system having the upgraded version even when the version upgrading program stored in the flash memory 42 is erased for
5 some cause. Even when the version upgrading program is erased due to power failure occurring in the course of its writing, the program having an older version stored in the auxiliary memory device 46 and the factory-shipped system program stored in the ROM 41 prevent the
10 system from misoperating or being incapable of starting up due to incomplete program updating into the flash memory 42.

(Fourth embodiment)

Then, description will be made of the fourth
15 embodiment with reference to FIGS. 6 through 9. The fourth embodiment is an example wherein an IRD (integrated receiver decoder) is used as a data processor.

FIG. 6 is a block diagram illustrating a
20 configuration of an IRD (integrated receiver decoder) for digital broadcasting preferred as a fourth embodiment. A broadcast wave is transmitted through a DVB (digital video broadcasting system), whereas images and voice are transmitted in accordance with MPEG2 of
25 ISO/IEC 61818-2 and MPEG2 of ISO/IEC 61818-3 respectively.

A reference numeral 610 represents a program run

portion which controls the IRD as a whole in accordance with a program and a reference numeral 606 designates a common bus which connects the program run portion 610 to each member.

5 A tuner 601 which receives a digital broadcast wave and selects a desired frequency as designated by the program run portion 610 provides a signal, which is subjected to demodulation, check for an error caused in a communication path and error correction in a
10 demodulation and error correction portion 602. Then, the signal is sent to a demultiplexer 603 which selects a stream having a desired program identifier (PID) out of multiplexed streams, and outputs the stream in a condition where it is divided into an image-voice
15 stream and a data stream which contains program information, program notification information and program data.

 The data stream is sent to a loader portion 607 and the program run portion 610. An image-voice stream
20 signal selected by the demultiplexer 603 is decoded by an AV decoder 604 into MPEG2 image data and voice data, which are reproduced into analog video signals and analog voice signals respectively by a reconstruction and screen synthesis portion 605 and output.
25 Furthermore, the reconstruction and screen synthesis portion 605 is cable of synthesizing an EPG screen, an operation screen or the like as designated by the

program run portion 610 and providing them as video signals.

A reference numeral 611 represents an input portion which transmits user's operations to the
5 program run portion 610 as input data from keys and a remote controller. A power switch and an OK key are included in this input portion 611.

A reference numeral 607 designates a loader portion which selects predetermined data from the data
10 stream, thereby performing hardware storing control, software information storing control and program storing control. A reference numeral 608 denotes a hardware information storing portion for storing hardware information such as a manufacturing company
15 and a model number which are not rewritten. A reference numeral 609 represents a software information storing portion for storing a software version number from the loader portion 607 which is rewritten into a downloaded version by a program updating work.

A reference numeral 612 designates a memory
20 control portion which controls a nonvolatility program storing portion A 614, a nonvolatility program storing portion B615 and a work area RAM 616. The nonvolatility program storing portion A 614 and the
25 nonvolatility program storing portion B 615 are composed of non-volatile memories which hold their contents even while they are not electrically energized

and provided as areas to store programs having different versions, and the memory control portion 612 determines, at a power on time, either of the programs stored in the nonvolatility program storing portions
5 which is to be executed. When a program to be updated from the loader portion 607 is downloaded and updated, the memory control portion 612 determines either of the nonvolatility program storing portions into which the program is to be written. The work area RAM 616 is
10 used by the program run portion 610 as a work area RAM during execution of a program.

A reference numeral 613 is a display device which uses a liquid crystal panel, a plasma panel or the like, and displays messages and operating conditions
15 such as "standby," "received program channel," "kind of network," "on program updating," "completion of program updating" and "failure of program updating" in pictograms, icons and characters.

A reference numeral 617 represents a timer portion
20 which can be set by transmitting a command from the program run portion 610 by way of the common bus 606 and is capable of notifying a predetermined time to the program run portion 610 when the predetermined time has elapsed.

25 A reference numeral 618 designates a power supply portion which can be set, even with a power switch turned off and a power supply cord is plugged in, in a

standby condition where power is supplied only to the program run portion 610 and the timer portion 617, whereas other portions are deenergized by transmitting a command from the program run portion 610 by way of the common bus 606 since the power switch is contained in the input portion 611. The power supply portion 618 is not set in the standby condition upon turning off the power switch but can be set in this condition after the program run portion 610 executes some processing.

When the power switch is turned on in the standby condition, the IRD is electrically energized as a whole and set in an operating condition. Reference numerals 619 and 620 represent lines to supply power from the power supply portion 618 to the program run portion 610 and the timer portion 617 respectively.

A program herein means a program which comprises a driver software such as an OS kernel or an MPEG driver or the like and an application software, etc. for EPG display screen or an operating screen.

Operations of the fourth embodiment will be described with flowcharts shown in FIGS. 7 through 9. Let us assume that an effective program is stored in the nonvolatility program storing portion A 614 and contents of the nonvolatility program storing portion B 615 are ineffective in an initial condition.

When the power switch is turned on in the standby condition (step S701), the memory control portion 612

reads out contents of the nonvolatility program storing
portion A 614 and the program run portion 610 starts
executing the contents of the nonvolatility program
storing portion A 614 (step S702). The program run
5 portion 610 adequately controls the tuner 601, the
demodulation and error correction portion 602, the
demultiplexer 603 and the AV decoder 604 by way of the
common bus 606, thereby setting each of the portions in
a condition that it is capable of receiving a broadcast
10 wave (step S703).

The IRD continuously receives a program designated
by a user who manipulates a key or a remote controller
on the input portion 611, and outputs a video signal
and a voice signal from the reconstruction and screen
15 synthesis portion 605 (step S704). When a PID which
indicates notification information, for example PID =
0040h, is received during reception of the broadcast
wave (step S705), the data, i.e., notification data
indicating "load data containing a manufacturing
20 company, a model number, a program version and a
program identifier (PID), as well as a transmitting
network number, a transport number, a transmission
start time and a transmission end time" is transmitted
by way of the demultiplexer 603 to the program run
25 portion 610 and the loader portion 607 which writes
"download data containing the program identifier (PID),
the transmitting network number, the transport number

and the end time" of the notification information into an empty area of the nonvolatility program storing portion A 614 by way of the memory control portion 612 (step S706).

5 The program run portion 610 compares the manufacturing company and the model number stored in the hardware information storing portion 608 with the manufacturing company and the model number contained in the sent notification information, and judges that a
10 program is to be updated in its IRD (step S707) and proceeds to the next step when the manufacturing company and the model number are coincident or intercepts the program updating when manufacturing company and the model number are not coincident (step
15 S711).

 Furthermore, the program run portion 610 compares the software version number stored in the software information storing portion 609 with the software version data contained in the sent notification
20 information, and judges that an updated program is to be sent (step S708) and proceeds to the next step when the version in the sent information is in advance or intercepts the program updating otherwise (step S711).

 By using transmission start time data, the program
25 run portion 610 reserves reception of an updated program by setting the timer portion 617 so that it informs a predetermined time a little earlier than the

transmission start time (step S709) to the timer portion 617.

Let us assume that the timer portion 617 informs the predetermined time a little earlier, for example 1
5 minutes, than the transmission start time (step S710). In the standby condition where electric power is not supplied to portions other than the program run portion 610 and the timer portion 617, the IRD is judged as inoperative (step S801), and power supply portion 618
10 turns on the power switch to supply electric power to each portion of the IRD (step S802). If the IRD is operating, the program updating is intercepted (step S711).

The program run portion 610 calls out the
15 notification information from the nonvolatility program storing portion A 614, sets the tuner 601 and the demodulation and error correction portion 602 using data of the transmitting network number and the transport number in the notification information, and
20 sets the demultiplexer 603 using downloaded data containing the program identifier (PID) (step S803). The program run portion 610 checks the tuner 601 for its receiving level and proceeds to the next step when the receiving level is larger than a predetermined
25 value (step S804) or intercepts the program updating otherwise (step S711).

The program run portion 610 reads out the

notification information containing the end time of
program updating from the nonvolatility program storing
portion A 614, checks whether or not another program
booking coexists before the end time (step S805) and
5 proceeds to step S806 when another program does not
coexist or intercepts the program updating when another
program coexists (step S711).

The program run portion 610 is set in a condition
where it does not receive a remote control key
10 operation from the input portion 611 to prevent the IRD
from misoperating due to an accidental input operation
during the program updating (step S806).

The display device 613 displays a pictograph "on
program updating" which notifies the user that a
15 program is going to be updated and a remote control key
operation is not received until the program updating
completes (step S807). The program run portion 610
starts receiving the program to be updated as reserved
(step S808). The program run portion 610 checks
20 received program data for transmogrification using the
CRC check and check sum (step S901) and proceeds to
step S902 when the program data is free from the
transmogrification or displays a pictograph indicating
failure of program updating on the display device 613
25 (step S907) and sets the IRD in the standby condition
when the program data has transmogrification and the
program updating is intercepted due to the occurrence

of an errors during program updating (step S906).

When the program run portion 610 judges that transmutation does not exist in the received program data in step S901, it writes the received
5 program data into the nonvolatility program storing portion B 615 in which the ineffective data is currently stored (step S902). Even if power failure or another cause makes it impossible to complete the writing of the program data into the nonvolatility
10 program storing portion B 615 at this step, the IRD can be started with the program stored in the nonvolatility program storing portion A 614 when the power switch is turned on once again since the nonvolatile program storing portion A 614 or the contents of the memory
15 control portion 612 are not changed at all.

After completing the writing of the program data, the program run portion 610 changes the memory control portion 612 so as to make access to the contents of the nonvolatility program storing portion B 615 (step S903)
20 and modifies the contents of the software information storing portion 609 into contents of the updated program version (step S904).

Upon completing the program updating as described above, the display device 613 displays a pictograph
25 indicating the completion of the program updating (step S905) and the IRD is set in the standby condition (step S906). When the power switch is turned on the next

time, the program which is written and updated in the nonvolatility program storing portion B 615 is loaded and executed.

5 Programs are stored alternately into the two nonvolatility program storing portions dependently on operating conditions of the IRD each time a program is updated.

(Fifth embodiment)

10 The fifth embodiment will be described with reference to the accompanying drawings.

The fifth embodiment has a configuration which is the same as that shown in FIG. 6, and operates in a sequence illustrated in flowcharts presented as FIGS. 7, 10 and 11.

15 Operations shown in FIG. 7 will not be described in particular since they are the same as those of the fourth embodiment which have been described above.

20 When all the portions of the IRD are electrically energized at a step S101, the program run portion 610 judges that the IRD is operating and proceeds to a step S102 or when all the portions of the IRD are not energized, operations of the program run portion 610 are the same as those at the step S802 and subsequent steps in FIG. 8 which are not described once again.

25 At the step S102, the program run portion 610 makes access to the notification information stored in the nonvolatility program storing portion A 614, set

the tuner 601 and the demodulation and error correction
portion 602 using the transmitting network number and
transport number contained in the notification
information, and sets the demultiplexer 603 using the
5 downloaded data containing the program identifier
(PID).

The program run portion 610 makes access to the
notification information the end time stored in the
nonvolatility program storing portion A 614, checks
10 whether or not another program booking coexists before
the end time (step S103) and proceeds to a step S104
when another program booking does not coexist or
intercepts the program updating when another program
booking coexists (step S711). Since programs can be
15 received at the same time when a network number and a
transport number of program updating are the same as
those of the program which is currently being received,
the program run portion 610 checks whether or not the
numbers are the same (step S104) and does not update
20 the program when the numbers are not the same (step
S711). This is because a video signal and a voice
signal may be recorded during reception of the programs
and imprudent program updating during the operation of
the IRD may make it unstable.

25 When the network numbers and the transport numbers
are judged as the same by the step S104, the program
run portion 610 is set in a condition where it does not

receive remote control key operation signal from the input portion 611 (step S105). The display device 613 displays a pictograph "on program updating" which informs the user that a program is to be updated and
5 the program run portion 610 does not receive a remote control key operation signal until the program updating completes (step S106).

The program run portion 610 starts receiving the program to be updated as reserved (step S107). The
10 program run portion 610 checks whether or not transmogrification exists in received program data using the CRC check and the check sum (step S111) and proceeds to a step S112 when the transmogrification does not exists or intercepts the program updating when
15 the transmogrification exists, whereby the display device 613 displays a pictograph indicating "failure of program updating" due to an error occurring during the program updating (step S119) and the IRD is set in the standby condition (step S118).

20 When the transmogrification does not exist, the program run portion writes received program data into the nonvolatility program storing portion B 615 in which ineffective data is currently written out of the two nonvolatility program storing portions (step S112).
25 Upon completing the writing, the program run portion 610 is ready for receiving the remote control key operation signal from the input portion 611 (step S113)

and the IRD is set in the usual operating condition.

When the user turns off the power switch on the input section 611 to terminate the operations of the IRD (step S114), the program run portion 610 changes the memory control portion 612 so as to make access to the contents of the nonvolatility program storing portion B 615 (step S115) and modifies the contents of the software information storing portion 609 into contents of the updated program version (step S116).

Upon completing the program updating, the display device 613 displays a pictograph indicating the completion of the program updating (step S117) and the IRD is returned to the standby condition (step S118). When the power switch is turned on the next time, the updated program which is written in the nonvolatility program storing portion B 615 is loaded and executed.

(Sixth embodiment)

The sixth embodiment is configured to resume a preceding program version when the user feels that he cannot use an updated program as described in the fourth or fifth embodiment conveniently or make it familiar with himself.

FIG. 12 shows a setting screen to modify a program to be started up. This screen is synthesized by the reconstruction and screen synthesis portion 605 under control by the program run portion 610 on the basis of an instruction made by the user on the input portion

611.

When the users issues an instruction OK by
operating a remote controller or a key on the input
section 611 in response to a question "Change program
5 ?," the memory control portion 612 modifies settings so
that a program is to be loaded, at a power on time,
from the nonvolatile program storing portion different
from the nonvolatile storing portion which stores a
program currently being executed. Another program is
10 loaded and started up when the power switch is turned
on once again after it is turned off.

By repeating these operations on the setting
screen, the user can select and execute two kinds of
programs.

15 It is needless to say that a program is not
updated when it is requested to update a program having
a version which is newer than that of a program
currently being updated but is the same as that of a
program stored in the other nonvolatile program storing
20 portion.

As understood from the foregoing description, the
present invention makes it possible to prevent a system
from misoperating due to incomplete program updating.

Furthermore, the present invention makes it
25 possible to prevent a data processor from operating
unnaturally due to program updating which is made while
it is operating.

Moreover, the present invention makes it possible to selectively use a plurality of programs as designated by the user.